COSC 3P71 Fall 2024: Assignment 2

Instructor: Reginald McLean: rmclean@brocku.ca
Teaching Assistants: Zachary McGovarin, Tyler McDonald, Michael Eshun, Rasa Khosrowshahli

Available Date: November 3rd, 2024
Due Date: November 24th, 2024, 11:59pm

Goal: Use a genetic algorithm (GA) to provide possible solutions to the described problem. Prepare a report outlining the performance of the GA for the given problem, including the experimentation needed to empirically choose parameters of the GA.

Languages: Any programming language which will compile and/or run on the lab computers.

Tasks:

1. Implement a genetic algorithm (GA) as outlined below for the class scheduling problem.
2. Empirically determine the hyperparameters (i.e. crossover rate, population size) through a number of experiments using the GA, collecting data on the performance of various hyperparameter configurations.
3. Prepare a report outlining the results of these experiments.

Genetic Algorithm Steps

1. Read in problem instance data
2. Set GA parameters (crossover rate, mutation rate, population size, etc)
3. Create a random initial population
   For 0..MaxGenerations; do:
   1. Evaluate fitness of individuals in the population
   2. Select a new population using selection strategy
   3. Apply Crossover
   4. Apply mutation

   End For

Genetic Algorithm Components:

1. Initial population initializer: A function that creates a population of some size of randomized solutions to the problem
2. Chromosome: the way to encode the solution to the problem
3. Reproduction/Selection Mechanism: Tournament Selection. Choose k (k=4) individuals to reproduce, select the best two to perform crossover
4. Crossover: Given the selected individuals, a crossover creates two offspring. Implement the GA using
   a. Uniform Crossover; and
   b. A crossover of your choice (ie 1 point, 2 point, ordered crossover)
5. Mutator: Given an individual from the population, create a mutated version of the individual

6. Fitness evaluation function: a function that receives a solution to the problem and returns a single scalar value that represents the quality of that solution
7. User parameters: population size, maximum number of generations, probabilities of crossover & mutation

BONUS:
For a 2% bonus on this assignment: in your experimentation, implement your own innovative idea. This could be a different initial population representation and creation strategy, a different selection scheme, a different (third) crossover not discussed in class, etc.

Assignment Details

The university class scheduling problem involves assigning courses to rooms and time slots while satisfying multiple constraints. This is a complex combinatorial optimization problem that is well-suited for solving using genetic algorithms.

Problem Components

1. Courses
    a. A unique name/identifier
    b. An assigned professor
    c. Number of enrolled students
    d. Duration (in hours)
2. Rooms
    a. A name/identifier
    b. Maximum capacity
3. Time Slots
    a. Day of the week
    b. Hour of the day
    c. Typically excludes lunch hours and after-hours

In order to schedule each of the courses, we can create a schedule chromosome representation as follows:

Chromosome = [(course_index, room_index, timeslot_index), (course_index, room_index, timeslot_index), …]

Where the length of the chromosome allows us to represent every class in this manner.

Example with this chromosome:
Let's say we have **three** courses:
1. Math 101 (Prof: Dr. Smith, Students: 28, Duration: 3 hours)
2. Physics 301 (Prof: Dr. Smith, Students: 56, Duration: 1 hour)
3. Chemistry 201 (Prof: Dr. Smith, Students: 38, Duration: 3 hours)

With **two** available rooms

1. Large Lecture (Capacity: 100)
2. Medium Lecture (Capacity: 75)

And the following **seven** time slots
Monday: 9:00 10:00 11:00 13:00 14:00 15:00 16:00

Then a randomly initialized chromosome would be:
[(0, 1, 4), (1, 0, 3), (2, 0, 0)] == [(Math 101, Medium Lecture, Monday 14:00), (Physics 101, Large Lecture, 13:00), (Chemistry 101, Large Lecture, 9:00)]
Where each component of the chromosome is a 3-tuple indicating (class index, room index, time slot index) and the length of the chromosome is equal to the number of classes that need to be scheduled.

Fitness Function

We have a few constraints that decide the fitness of our solution:
1. A course with X students cannot use a room with Y seats if X > Y.
2. A room booked at X:00 cannot have class B booked into the same room within the duration of class A.
3. Similarly, a professor cannot teach two classes that overlap with each other.

When evaluating a chromosome, you should keep track of the number of conflicts that the chromosome has and return the fitness of that chromosome as 1 / (1 + number of conflicts) where we want to **maximize** this fitness value. **The pseudocode of the fitness function can be found in this file.**

In our previous example of the randomly initialized chromosome, we would get a fitness value of:
Num_conflicts = 0
[(0, 1, 4), (1, 0, 3), (2, 0, 0)]
Chromosome[0] = Math 101, Medium Lecture, 14:00
Math 101 28 students < medium lecture 75 seat capacity
Math 101 time 14:00 in Medium Lecture: no conflicts with other classes in Medium Lecture from 14:00-17:00
Math 101 professor Dr. Smith: no conflicts with other classes they teach

Chromosome[1] = Physics 101, Large Lecture, 13:00
Physics 101 56 students < Large lecture 100 seat capacity
Physics 101 time 13:00 in Large Lecture: no conflicts with other classes in Large Lecture from 13:00-14:00
Physics 101 professor Dr. Smith: no conflicts with other classes they teach

Chromosome[2] = Chemistry 101, Large Lecture, 9:00
Chemistry 101 38 students < Large lecture 100 seat capacity

Chemistry 101 time 9:00 in Large Lecture: no conflicts with other classes in Medium Lecture from 09:00-12:00
Chemistry 101 professor Dr. Smith: no conflicts with other classes they teach

Fitness = 1 / (1 + num_conflicts) = 1.00. This is an optimal solution to the problem.

NOTE: If struggling to implement the fitness function, use this example and the pseudocode as a guide. If you can generate a value of 1.00, manipulate the given chromosome to purposefully introduce a conflict.

Problem Data
The problem data is available at this Google Drive link.

Experimental Analysis

Run your GA to compare the performance of the two crossover operators mentioned above by using the following parameters (and include elitism in all cases):

1. Crossover rate = 100%, Mutation = 0%
2. Crossover rate = 100%, Mutation = 10%
3. Crossover rate = 90%, Mutation = 0%
4. Crossover rate = 90%, Mutation = 10%
5. Determine your own best settings

For elitism, first consider an elite strategy where only the best chromosome is replicated to the next generation. Next consider replication criteria where a certain percentage of individuals determined empirically (no greater than 10%) are allowed to replicate (include chromosome of best fitness value for this replication)

For each experiment mentioned above, run your GA at least 5 times on both the t1 and t2 test cases located in this zip file.

Output the following to a file or standard output:

a) All GA parameters, including random number seed

b) Per each generation: best fitness value, average population fitness value

c) Per each run: best solution fitness and its corresponding best solution chromosome

For your analysis, compute for the multiple runs: average of best fitness per generation and average population fitness per generation. Using a graph drawing tool such as excel, plot well labeled graphs for experiment 1, above including experiment 2 (if done). Types of graphs you plot for experiment 2 depend on your incorporated idea, if any. Feel free to experiment with different crossover and mutation rates. You will set your own Pop-Size and generation size

Experiment Tracking Tip

A lot of ML projects now use MLOps tracking tools (i.e. Weights and Biases in Python or MLFlow in Java) to handle the organization and tracking of experiments. Please feel free to use those tools to make tracking your data simpler.

Assignment Report

Once your data is collected and your analysis is complete, you will prepare a summarized report of your findings using the IEEE conference format introduced to you during tutorial. IEEE format details are found at:
https://www.ieee.org/conferences/publishing/templates.html

If you download the LaTeX template, you can use something like overleaf.com – an online LaTeX editor that simplifies the process of compiling & producing conference PDFs.

The report should have each of the following sections and each section should address the listed points.

- Introduction

BRIEFLY introduce the concepts and topics discussed in the report.
Precisely define the problem you implemented and explain why its solution is important.

- Background

This section should explain the algorithms used in the report (pseudo code is helpful) and may provide other information which you feel will be relevant to someone trying to understand your results. The goal of a background section is that if someone who did not know anything about GAs read your report, they would have enough details to understand what you did for your experiment.

- Experimental Setup

This section should provide enough information about your experiments to allow someone else to duplicate your results.
This should include algorithm parameters used, the crossover and mutation operators used, and any other relevant implementation details.

- Results

This section should summarize your findings. For your multiple runs compute the average of the best fitness per generation and the average population fitness per generation. Using a graph drawing tool such as excel, plot well labeled graphs for your experiments. Also include summary tables describing the fitness of your final solution. Summary statistics such as min, max, mean, median, and standard deviation should be included in your tables.

Tests for statistical significance would also be appropriate, for example T-Tests or Mann Whitney U tests. Explain your graphs/data in detail and emphasize the similarities and differences between different algorithm configurations.

- Discussions and Conclusions

This section should provide a BRIEF summary of what experiments you performed and the results you observed.
Following this BRIEF summary, you should discuss your opinions regarding your results and what conclusions you've arrived at.
This could include issues like which crossover performed better. If more than one mutation type was tried, which one performed better. If you included local search, did it help? How did the choice of GA parameters affect the final outcome etc.?

- References

List your sources here. The text of the report should contain references to your sources.
This report is very important, so be sure to include it. Start early, gathering the data and doing the experimental analysis will take much more time than coding the assignment.

Submission Details

- Your electronic submission should include:
- Your source code
- An executable
- Instructions for compiling/running your program and changing parameters
- The data you've generated for your report
- Your written Latex report as a pdf or doc file.

Submission will be done electronically using Brightspace. Any questions regarding submission can be directed to Zachary at zm19hc@brocku.ca

Please note that the virtual COMMONS is available to all students at Brock. You are not required,but if you prefer to (or need to) you can use the virtual COMMONS instead of a personal computer. Machines in the virtual COMMONS have IDEs for Java, Python, and C#.
Any questions/concerns regarding the grading of any assignment MUST be raised within 7 days of graded assignment hand-back. In this case, please send your concerns/questions to Zachary at zm19hc@brocku.ca.

To better serve you, please don't send multiple queries on the same topic to the course coordinator, Professor and other TAs. The course coordinators will be the point of contact for any such queries and the Professor will receive them all at once from them.
Feel free to use any language (with reason) as long as it can be opened and executed on the lab computers. Examples include Java, C#, C++, and Python. No matter your choice of language, ensure you have provided sufficient comments such that your program can be understood by the markers. At a minimum, include a comment describing each function/method and class/module. Unity projects are not allowed for this assignment.

This assignment is to be completed individually. Plagiarism detection software will be applied in this course for all submitted work. Additionally, a number of assignments will be randomly selected, and the authors will be asked to explain their code and submitted documents.